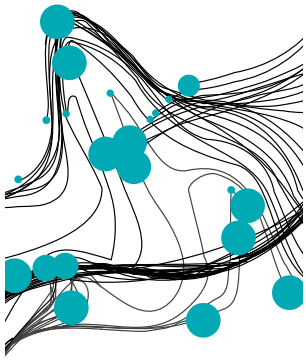
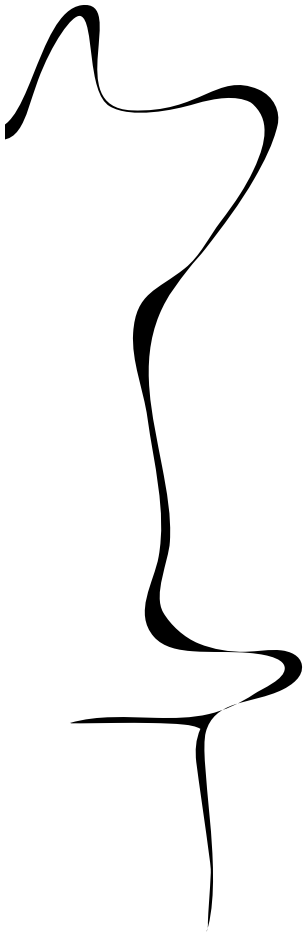


RAM ROBOTICS AND MECHATRONICS



# SOFTWARE INTEGRATION OF ELECTRONIC SPEED CONTROLLER (ESC) FOR AN UNMANNED AERIAL ROBOT

H. (Hussein) Osama Hussein Abdelrahman

BSC ASSIGNMENT

**Committee:**
A. Franchi, Ph.D HDR
ir. H. Das
Y.A.L.A. Aboudorra, MSc
dr. V.V. Lehtola

July 2021

040RaM2021
Robotics and Mechatronics
EEMathCS
University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands

UNIVERSITY OF TWENTE. | TECHMED CENTRE

UNIVERSITY OF TWENTE. | DIGITAL SOCIETY INSTITUTE

# Contents

**1 INTRODUCTION** **4**

**2 Project Summary** **5**

**3 Background** **6**
3.1 UAV System . . . . . 6
3.2 PID controllers . . . . . 7
3.3 Motor controller protocols . . . . . 7
3.4 Software architecture and Simulation . . . . . 8

**4 METHODOLOGY** **10**
4.1 Overall System . . . . . 10
4.2 Hardware . . . . . 13
4.3 software . . . . . 14
4.4 Simulation . . . . . 16

**5 RESULTS and DISCUSSION** **17**
5.1 Tests on motor . . . . . 17
5.2 Simulation results . . . . . 18
5.3 Discussion of Results . . . . . 19

**6 CONCLUSION AND RECOMMENDATIONs** **21**
6.1 Conclusion . . . . . 21
6.2 Recommendations . . . . . 21

**Abstract**

In this paper,the software integration of Dshot protocol was implemented on a hex rotor Unmanned Aerial Vehicle (UAV). Dshot protocol is a digital form of communication between the motor controller and the electronic speed controller (ESC). The KISS ESC is used since it can run Dshot protocol, and it is connected to the Teensy microcontroller running the motor control program. The implementation was done using Genom3, a real-time software architectures design tool. The integration will be done by modifying an existing Genom3 component that can communicate with the hardware connected to allow serial communications with the teensy. The data was collected by testing the Genom3 component on a single motor using the functions and services implemented in the modified Genom3 component.

# 1 INTRODUCTION

Unmanned Aerial Vehicles (UAV) are regarded as a modern invention for most people, and it is easy to understand why. If we go back years ago and think about the idea of buying a personal drone or having a drone deliver packages, it would feel more like a science-fiction idea. The fact is that the first reported UAV was developed by Joseph-Michel and Jacques-Étienne Montgolfier who developed an unmanned hot air balloon back in 1783[1].

A UAV (also called a drone) is an aircraft that has no human pilot or any other passenger. A UAV can be fully autonomous and fly on its own, but usually it is also manually piloted by a human. UAVs are becoming more popular everyday for their various applications in many fields including aerospace, military, and many scientific fields. UAVs are extremely beneficial since they can navigate without a human pilot, which means they can access places that are difficult to access if a human pilot is needed. This explains why the use of UAVs is rising over time as over the years, the number of UAV publications per year is rapidly on the rise.

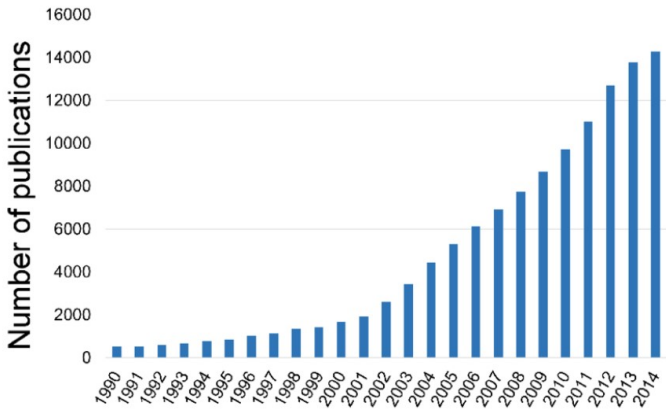



Figure 1: UAV publications per year [2]

The UAV system is a complex system involving a lot of different hardware and software components. A complex system means there are many possible implementations using different hardware, software, and different protocols to implement. In this project, the main goal is to implement Dshot [3] protocol on a KISS ESC [4], and utilize it for a hexcopter UAV. The hexcopter UAV is running many Genom3 [5] components, so Dshot will be implemented by modifying the components already present. The modification will be done since to implement Dshot protocol, a KISS ESC [4] with Teensy 4.0 [6]will be used to run the motor instead of the ESC used with analog protocol.

## 2 Project Summary

As mentioned previously, the main goal of this project is to implement Dshot protocol using Genom3. This is the software part of the project but since the actual hexcopter UAV has to be physically built, another student will be working on the hardware part of the project. The task the other student will have is to build the hexcopter UAV that is running on Dshot, hence the student will have to build the UAV and conncet all the pieces together inculding a mount for the six KISS ESCs along with the teensy for the Dshot part. Also, the power management of all these ESCs Will be handled to make sure the motors and ESCs receive the optimal power to operate and fly the UAV.

# 3 Background

## 3.1 UAV System

In order to build a UAV, both hardware and software have to be taken under account. The hardware can be divided into four subsystems: actuators, ground control, flight controller, and motor controller[7]. For the software part, the integration will be done using Genom3 which creates modules and services running on c and c++[5].

**Actuators** The actuators consist of the motor and the propellers. The number of motors and propellers in a UAV depends on the type of the UAV itself. Since a Hexcopter UAV is being built, it will have six motors and six propellers one for each motor. The motors are DC brushless motors (BLDC) [8]. For a UAV, electrical propulsion is more suitable since it produces less heat, less noise, and no emissions. This is useful for the applications of the UAV, which can include stealth missions for example[9].

BLDC motors have a permanent magnet and three phases of driving coils with a sensor that tracks the rotor position. The BLDC motor is commutated electronically instead of by brushes, thus eliminating the need for brushes which can wear off with time[8]. BLDC works with an ESC which controls the BLDC motor velocity by creating the appropriate magnetic field so that the motor rotates as desired [10].

**Ground control** The ground control is the center that allows human navigation of the UAV. The ground control can include different systems like computer, telemetry, video capture card and aerials for the control and data links to the UAV . The user has a Graphical User Interface (GUI) that shows the status of the UAV, flight information, virtual cockpit, map screen for navigation, etc. These features depend on which Open-Source Project is used, since each has its own GUI with its own features.

**Flight controller** The flight controller consists of the flight avionics and a radio receiver for communication with ground control. The flight avionics includes sensors, I/O pins, and a processor. The sensors are typically a gyroscope, accelerometer, Global Positioning System (GPS), barometer, and a magnetometer.

The barometer and GPS are used to determine current time, location, and altitude. The angle of the UAV is obtained from the gyroscope and accelerometer. The problem with the accelerometer and gyroscope is that the accelerometer has high frequency noise in its measurements and the gyroscope contains low frequency noise, since the gyroscope's output needs to be integrated to obtain the angle which adds drift to the measurements. In order to calculate the angles of the UAV, a complimentary filter is implemented which is a combination of a high pass filter for the gyroscope and a low pass filter for the accelerometer.

## 3.2 PID controllers

PID controllers are the most common control algorithms, it is used to regulate and stabilize different process variables by the use of closed-loop feedback mechanism. PID controllers are popular due to their robustness and their relative simplicity. PID stands for the three basic coefficients: proportional, integral, and derivative. The closed loop is just the error of the input reference point and the output measured data. The proportional component depends only on the error value, it is not affected by previous values or changes. The integral component adds up the error value to drive the steady-state error to zero by time. The derivative component depends on the rate of change of the error, thus it controls the response speed of the controller.

This project includes a position, attitude, and motor controllers. The attitude and the position controller systems control the attitude and position of the UAV. These controller take the current state of the UAV as input along with the reference, which depends on the desired way point or position, and output is a control signal consisting of the desired velocity or thrust required to achieve the desired reference position.

The motor controller takes the required thrust from the attitude and position controllers and the actual velocity of the motor as inputs, and will output the required velocity of the motor to achieve the required velocity from the attitude and position controllers.

## 3.3 Motor controller protocols

Since the motor controller sends the required velocity of the motor to the ESC, there has to be a communication protocol between the motor controller and the ESC. There are different communication protocols implemented, but they can be divided into two groups analog and digital.

**Analog** The most common Analog communication protocol is the Pulse Width Modulation (PWM) method. The PWM signal is an analog signal where the duty cycle of the pulse sent to the ESC is used to determine the rotation of the motor.
The PWM protocol is the oldest protocol and over the years more protocols which are based on the PWM protocol were developed. This is due to the fact that the PWM method has a slow refresh rate, thus this protocol could be too slow depending on the application of the drone. Nevertheless, the refresh rate is not the only parameeter that is important for drones application, since parameters like Proper tuning, motor matching, filtering and vibration reduction can be more important than a slightly faster refresh rate[11].

OneShot 42 and OneShot 125 are both protocols that are related to the PWM method. OneShot 42 sends data packets every 42-84 $\mu s$. On the other hand, OneShot 125 sends every packet every 125-250 $\mu s$. The old PWM method sends a packet every 1000-2000 $\mu s$, thus one shot 42 and OneShot 125 are clearly much faster than the PWM method[12].

Another analog protocol is MultiShot, which send a packet every 5-25 $\mu s$, making it 20 times faster than the PWM protocol.[12]

**Digital** In general Analog signals have disadvantages like inaccuracies and corruption due to noise. Also, analog signals are usually limited in frequency and performance speed. Thus a digital signal can solve these problems, which is done by using D-shot protocol.

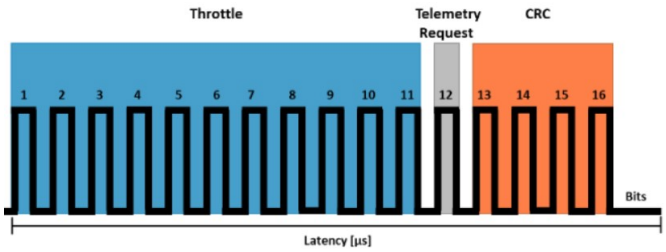



Figure 2: Dshot Packet [3]

D-shot protocol involves sending a packet of 16 bits, as shown in figure 2. The first 11 bits represent 1 pulse each with a duty cycle that depends on the value of the bit.The last 4 bits are the cyclic redundancy check (CRC) bits which is another advantage of using Dshot, since this minimizes the bit error.

## 3.4 Software architecture and Simulation

Robots and autonomous vehicles have a complex system consisting of many software pieces, where many different tasks have to be implemented with different requirements and time constraints. For example, some task will be running real-time while other tasks will be doing offline data processing. In order to handle the software integration of this complex system, component-based software architectures is used.

**Genom3** The Generator of Modules (GenoM) is a tool to design real-time software architectures dedicated to systems such as autonomous mobile robots or satellites. Genom offers components which are running specific functions and algorithms.

Dealing with functions with different real-time constraints and algorithm complexities requires a coherent way of handling starting, ending, pausing, and error handling of functions and the interfaces and the data flow between the functions and components. Genom does this by creating a general template that that is the same for all components to make sure all the components have the same behaviour [5].

Genom3 allows the creation of a component by writing the component description, then a template is generated automatically based on the component

description. The component description file includes details about the component itself which is just information about the component and the author. Next are the details of the input and output ports, which connect this component with the other components. After that comes the internal data structure (IDS) which are the variables that are shared within the component, so it is shared by the different services and tasks running. Next comes a description of the three possible types of any services which are attribute, activity, and function. Also then the description of the execution tasks which are responsible for executing the activities of the component. Execution taks can be periodical or running continuously, depending on the service. Different execution tasks can also be given different priorities, depending on the application.
The algorithms can be implemented by adding the c++ or c code on codels folder produced with the template generated.

**Gazebo** Robot simulation is a vital part of every robotics toolbox since it makes it possible to test the algorithms developed in a simulated real environment. Gazebo offers this by accurately simulating populations of robots in different environments. One of the advantages of using Gazebo is that it allows custom plugins for robots and environmental control, these plugins allow access to Gazebo's API [13]. Therefore, a plugin can be used to emulate the tk3 - mikrokopter software[14], the main software which includes the flight controllers and the communication with the hardware. This means that all hardware functions implemented can be tested in a safe simulated environment to debug and test the performance of these functions.

# 4 METHODOLOGY

## 4.1 Overall System

Before the overall system is explained, the Genom3 components used will be explaned. The Genom3 components running on the Odroid are part of TeleKyb3 platform. TeleKyb3 platform is an open-source software framework for the development of UAVs [14]. Telekyb3 provides components for state estimation, trajectory planning, processing, and tracking.

**TeleKyb3 Genom3 components**

- uavpos: This component implements a three dimensional position controller. It has the following inputs:
  - The current wrench exerted by the UAV according to the propeller measurements,
  - State of the UAV. This includeds all the data that corresponds to the sate of the UAV including velocity, angular velocity, acceleration, position, and attitude. All these measurements of the state are three dimensional, hence the data is given for the three axis x,y, and Z.
  - Last input is the reference data. This is the other input to the controllers, since the reference and the state are what determines the error for the controller. The reference data includes the desired velocity, accelration, position, etc. Moreover, again the data is for all three axis[14].

  The uavpos component has one output UAV-control, which is the Attitude/thrust control input. This is the input of the next component, uavatt.

- uavatt: this component implements the attitude controller. The component takes the output of the uavpos component as input. The inputs of uavatt are :
  - UAV_control.
  - State of the UAV.
  - rotor_measure. This is the measurements of the motor, so it includes information like velocity, throttle, consumption, and energy level.

  Outputs of uavatt:
  - Wrench measurement. This is an input for the uavpos component
  - rotor_input. This output contains an array of desired velocity and thrust for each motor [14].

- maneuver: The main task of the maneuver component is to implement take-off,landing, and moving to a specific waypoint-based trajectories. It takes as input the state of the UAV and outputs the reference which is the input of uavpos component.
- mrsim: mrsim is a multirotor simulator which emulates the telekyb-mikrokopter communication protocol and implements the telekyb3-mikrokopter features. Also, mrsim provides a simulation of a gps/motion capture device using the GPS output port.
- mikrokopter: mikrokopter is the embedded autopilot running on the Paparazzi Chimera Board. There are two main programs in mikrokopter, mkbl which is the BLDC motor speed controller with PWM or velocity control.Besides mkbl the other program is the autopilot program mkfl[14].
- rotorcraft: rotorcraft is the component responsible of communicating with all the hardware connected to the Odroid. Rotorcraft receives an array of desired rotor velocities and thrust sent by attitude and position controllers and sends them to the connected hardware [14]. Rotorcraft has only one input which is the rotor_input coming from the uavatt component. The outputs of rotorcraft are :
  - IMU data which is needed for the uavposs and uavatt components.
  - Magnetometer measurments
  - rotor_measure.

  The main goal is to implement Dshot by adding the Teensy 4.0 to the system, so the modifications will all be done in this component since it is the component responsible of communicating with the ESC.

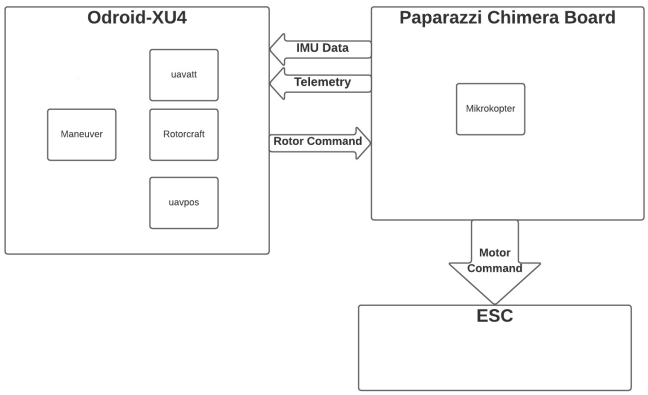



Figure 3: Initial System without Dshot protocol

The overall system can be split into hardware and software. The hardware is running software components that encapsulate all the algorithm which are running the UAV. Figure 3 shows the overall system, with the outer box represents the hardware and the inner small boxes each represent a Genom3 component. The system starts with the Odroid mini PC which is running the high-level controllers (Flight controller), which are the position and attitude controllers. The output of these controllers is sent as rotor command to the Paparazzi Chimera board, which is running Genom3 components that communicate with the ESC. These Genom3 components are running the low-level motor controller and their output is sent to the ESC.

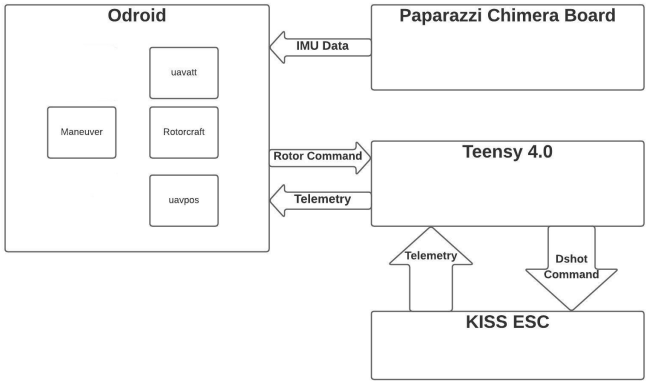



Figure 4: New system with Dshot protocol using Teensy

**New implementation of the system** Since Dshot will be implemented and the Teensy 4.0 will be used, a new system was made. Figure 4 Shows the new implementation of the system. The main difference is that the Paparazzi Chimera board is now only used to obtain Inertial Measurement Unit (IMU) data, which is needed as input to the flight controllers. The rotor command (output of flight controllers) is now sent to the Teensy 4.0, which is responsible for the communication with the KISS ESC and this communication is done using Dshot protocol.

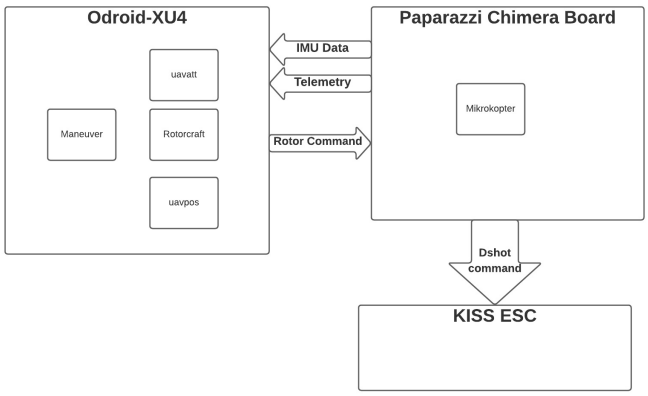



Figure 5: Alternative system with Dshot protocol using the Paparazzi Chimea Board

**New implementation of the system** The alternative system is a simplified version of the previously discussed system with the Teensy. The main advantage of implementing this system is that it requires less hardware and thus saves up power and space. In order to implement this system, a Dshot protocol must be implemented using the Paparazzi Chimera Board [15] which has the STM32F767 micro controller. The STM32F767 is able to run the Dshot protocol since it has 2 DMA controllers, which are used for the Dshot protocol.Nevertheless, the implementation with the Teensy was chosen since it would not require creating a program to run Dshot protocol using DMA. This is due to the fact that there is an open-source implementation already available for Dshot protocol for the Teensy, thus saving time and complexity given the timeline of the project.

## 4.2 Hardware

**Odroid** The Odroid is a powerful single-board computer capable of running various flavours of Linux and Android, offering a decent open source support. The Odroid used is ODROID-XU4 which implements interfaces like USB 3.0 and Gigabit Ethernet, thus the ODROID-XU4 offers fast data transfer speeds [16].The Odroid will be running the flight controller Genom3 components (attitude and position controllers).

**Paparazzi Chimera Board** Paparazzi is an open-source autopilot system that comes with the airborne processor with its sensors, autopilot software, a ground control station, the communication protocols between components and a simulation environment[15]. The Paparazzi Chimera board is used to obtain attitude and position data for the flight controllers by the use of use of the following in-board sensors [17]:

- 9 degrees of freedom IMU (MPU-9250): The IMU includes a 3 axis Accelerometer ,3 axis Gyroscope, 3 axis Magnetometer that are used to determine to the pitch, roll, and yaw of the UAV.
- Barometer/Altimeter:A High resolution Integrated digital pressure sensor which is used to determine altitude.

**Teensy and KISS ESC** The Teensy is a USB-based microcontroller development system that comes in a relatively small size. The Teensy 4.0 used has ARM Cortex-M7 processor which runs at 600 MHz [6], making it very suitable for the fast real-time processing needed for the UAV. The Teensy 4.0 will be running the motor controller, where the output of this motor controller will go to the KISS ESC.

The KISS ESC is a 32bit brushless Motor Controller. It offers many features such as current measurement and limiting, temperature protection, and telemetry[4], thus making it very suitable for multi-rotor pilots in general.

## 4.3 software

**Rotorcraft modifications** The rotorcraft component has to be modified to allow connection with Teensy and to control the motor. The main aim is to replicate all the rotorcraft services and functions that controls the old ESC, thus none of the functionalities of rotorcraft will be lost after switching to Dshot protocol.

The first step is to add a service that initiates the serial port and start the serial connection with the Teensy, and a service to end this connection. This was done by creating two activities, connect_teensy and disconnect_teensy. connect_teensy and disconnect_teensy take as input the serial number of the teensy, so by making the user input the serial number the service can work with any teensy as long as the correct serial number is entered. The execution task of connect_teensy and disconnect_teensy is task comm, which is a task that is running continuously, this is to make sure that the connection with the teensy is always running.

The next service added is activity start_teensy. start_teensy has the purpose of spinning the propellers at the minimum velocity of 450 rpm. The execution task of start_teensy is the periodic task main which runs at a frequency of $1000Hz$ since it does not need to be running continuously, but it should run at the frequency of the whole system. start_teensy works by sending a low reference rpm below or equal to the minimum velocity (450 rpm) to each of the motors every time it is called.

After initializing the serial connection and spinning the propellers at the minimum velocity, the service that should be added next is activity servo_teensy. This will make the propellers spin at the velocity requested by the flight controller output. Again, The execution task of servo_teensy is the periodic task main. For the propellers to move at the desired velocity of the controller, the rotor_input port (the output of the uavatt component) will be used as input to

the component. Then in the main codel rotor_input is input and by using the function read(self) which refreshes rotor_input with the latest data, the array "desired" is extracted. This array contains the desired velocity of each motor starting with the first motor until the 6th motor, since this is a hexcopter UAV. One step that needed to be done is multiply the values of the desired velocity by 60 since the values are in $Hz$ while the values sent tot eh ESC are in rpm.

Set_velocity_teensy is a function that allows the user to manually set a velocity to the motor. This can be done by creating a function which executes the task when called and thus is not linked to any execution task. The function takes the wanted velocity as input and sens that value as the refrence rpm to the ESC.

Another function created was set_gain_teensy, which sets the PID gain values manually, allowing the user to tune the controller for optimization. The function was made by creating an IDS variable, hence it is shared between the all the services. This is needed so that the other services know what PID gains to send to the teensy, otherwise the other survives will overwite the manually set gains with the old gains. The takes the three gains $K_P, K_I, and\ K_D$ as input and update the IDS variable accordingly.

Next comes stop_teensy, which is needed to stop all propellers. The activity stop_teensy takes no input and interrupts the activities servo_teensy and start_teensy since they are the activities responsible of moving the motor, and then sets the velocity to zero rpm.

Finally, the telemetry data should be received properly and allowed to be accessed by all the services. This is done by creating another ids which holds all the telemetry values from the ESC. The telemetry data include velocity, voltage, current, motor number, and error. An activity telemetry_teensy that is responsible for updating the telemetry all the time is created. Since it should be updating the telemetry all the time, it is linked to the comm task.

| Service name | main task | type of service | Execution task |
|---|---|---|---|
| connect_teensy | Initiate serial connection with Teensy | Activity | Comm (running continuously ) |
| disconnect_teensy | Release serial connection with Teensy | Activity | Comm |
| start_teensy | Move propellers at minimum velocity | Activity | Main (periodic) |
| servo_teensy | Move propellers with required flight controller velocity | Activity | Main |
| set_gain_teensy | tune PID coefficients | Function | None |
| Set_velocity_teensy | manually set velocity of propellers | Function | None |
| telemetry_teensy | Update telemetry data | Activity | Main |

Table 1: Summary of services implemented

**Dshot and Teensy implementation** The Teensy is running a program that implements Dshot communication with ESC, using Direct Memory Access (DMA). The advantage of using DMA is that it is faster, since it allows access to main system memory (random-access memory) without the need of the central processing unit. The program also is running a velocity PID control running at 500Hz implemented for up to 6 motors and receives telemetry data through UART[18].

The input data of the program are the velocity reference and PID parameters, and the output data are the telemetry data, error code and the Dshot command to the KISS ESC.

## 4.4 Simulation

Before running tests on the hexcopter UAV, the Genom3 components should be simulated to see how the whole system connects together and whether the result is a stable flying UAV. The simulation will be done by running all the Genom3 components on Gazebo. Since the new implementation of the system adds the Teensy to communicate with the KISS ESC and the motors, the simulation must be changed to fit the new system. This will be done by simulation the original system without the Teensy and use the data from simulation as input to the Teensy 4.0, thus making it a combination of simulation and real environment. The data that will be extracted from the simulation will be extracted from the output data of the flight controller and input to the Teensy, using the modified Rotorcraft component.

# 5 RESULTS and DISCUSSION

## 5.1 Tests on motor

**Serial connection test** The first step was to test connect_teensy and disconnect_teensy. This is done by calling connect_teensy and running the functions like start_teensy, if the functions run then connect_teensy worked. Then disconnect_teensy is called and again start_teensy is called to test if the serial connection was properly released, and that disconnect_teensy is working.

**Test on propellers** The next step is to test start_teensy which starts moving the propleers at the minimum speed, and set_velocity_teensy. First, start_teensy is called on MATLAB, and the propellers moved at the minimum velocity as shown in figure 6, where rpm_r is the rpm sent to the ESC and rpm is the actual rpm of the motor.

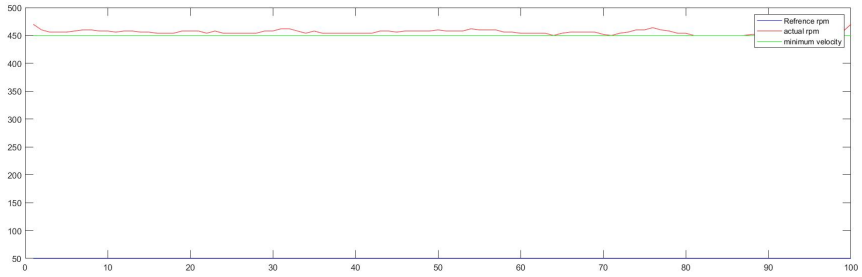



Figure 6: Start_teensy Test

Next, set_velocity_teensy was tested by creating a function in MATLAB that takes desired velocity and how long to move in seconds. The function then calls set_velocity_teensy on a loop depending on the time input in seconds. The results are shown are shown for velocity of 500,1000,2500 rpm in figure 7,8,9, respectively.

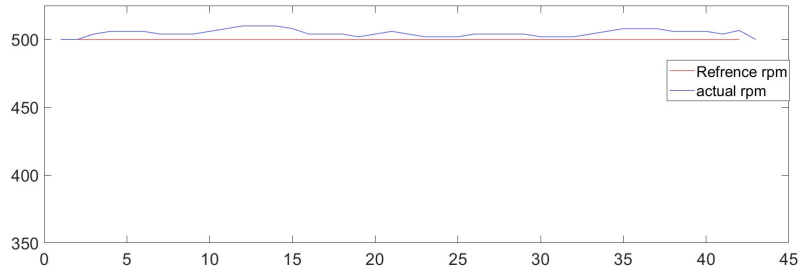



Figure 7: 500 rpm Test

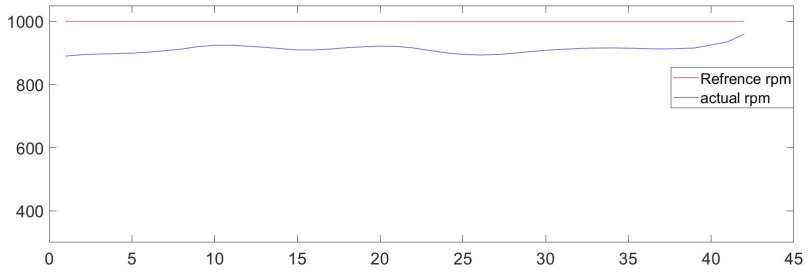



Figure 8: 1000 rpm Test

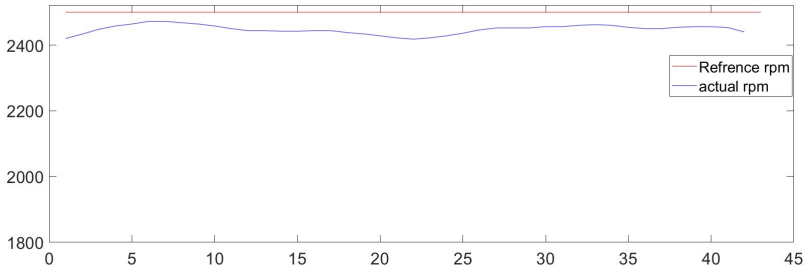



Figure 9: 2500 rpm Test

## 5.2 Simulation results

The simulation was run to test servo_teensy by using the output of simulated flight controller as input to the real motor. This was done by letting the UAV fly and observing if the motor moves identical to the simulation. The result

showed servo_teensy following the motors in the simulation since it moved and stopped identically to the simulated motors. The simulation telemetry data was successfully obtained using the log function available in rotorcraft component. The simulation is done by setting the UAV to take off to a certain height, and then the velocity data for each motor is observed. The velocity of each motor can be found in figure 10.

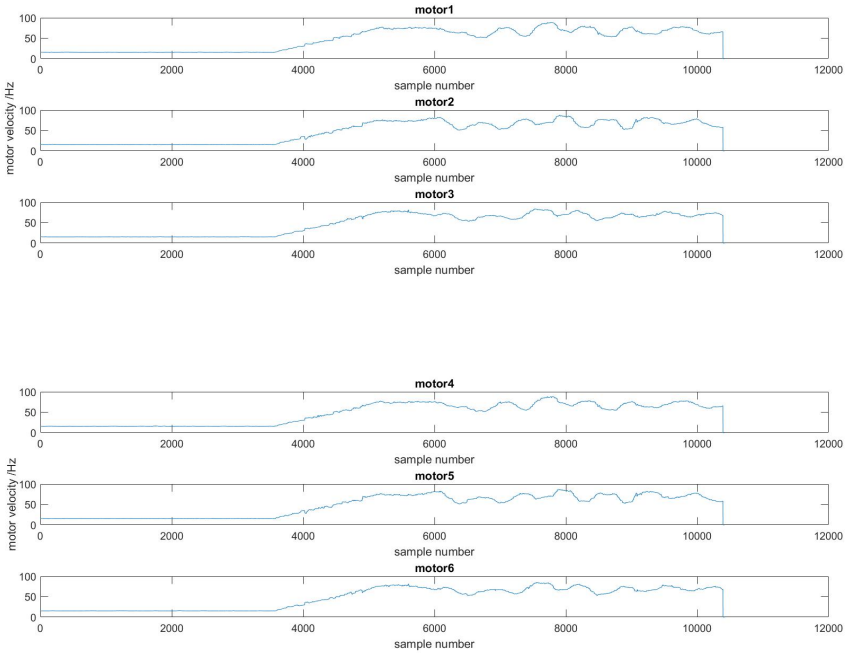



Figure 10: Simulation results

## 5.3 Discussion of Results

First connect_teensy was tested by calling it and testing if the functions will work (if serial connection has been established), and then discconnect_teensy was tested by calling it and again running a function to test if the serial connection has been broken. Both of the tests gave the expected results, hence the serial connection can be successfully initiated and released using the two services.

Furthermore, start_teensy and set_velocity_teensy were then tested on a single motor, by using the services to set specific rpm for the motor and reading the telemetry to determine the actual rpm of the motor. Figure 6 shows teh actual and refrence rpm and also the minimum velocity in rpm. The actual rpm is the current measured rpm of the motor, while the reference rpm is the rpm commanded to the ESC by the activity start_teensy. The minimum velocity is

450 rom, which means if any value less than that is sent as refrence, the motor will spin at 450 rpm. Thus, since start_teensy sends a refrence of 50, the actual rpm should be 450 as shown in figure 6.

For set_velocity_teensy, figures 7,8, and 9 show the result of sending rpm values of 500, 1000, and 2500, respectively. The results show the the actual rpm of the motor follows the rpm_r sent using the set_velocity_teensy function, indicating that the fuunction is working as expected since the actual rpm is following the reference rpm sent. The figres also show that for refrence rpm of 500 the actual rpm of the motor is very close to the refrence rpm, while on the other hand for rpm of 1000 and 2500 is slightly further away.

The simulation results in figure 10 show the velocity of the 6 motors during a take off to a height of 9 meters. The graphs show the velocity for each motor is not exactly the same as expected, which again shows the importance of mapping the values of the required velocity of each motor to the correct motor. This is why in the telemetry it is important to have the number of the motor to determine which motor do the values correspond to.

# 6 CONCLUSION AND RECOMMENDATIONs

## 6.1 Conclusion

In this project, the software integration of Dshot protocl on a hexcopter UAV was done using Genom3. Genom3 was used by creating different software components that carried out specific tasks and the Odroid XU4 single board computer was used to run all these Genom3 components. Dshot protocol was implemented by KISS ESC connected to a Teensy 4.0 which communicates with the odroid using one of the components running on the Odroid XU4 called rotorcraft. The rotorcraft component was modified to add the connection and communication with the KISS ESC using the Teensy 4.0. This was done by adding services that allow connecting and disconnecting to the Teensy, and services that are responsible of sending the desired velocity of the motors, either from the flight controller or by user input.

## 6.2 Recommendations

The services and functions connect_teensy, disconnect_teensy, start_teensy, and set_velocity_teensy worked as expected with no known flaws. servo_teensy was only tested for one motor and since the desired velocity is different for each motor and each value should be properly linked to the correct motor, servo_teensy can not be declared fully done. Also, there was no graphical representation of the desired velocity (from flight controleer) and the actual velocity of the motor for servo_teensy, which would act as proof that it is working as expected. For future attempts, a test on multiple motors should be done along with modifications on servo_teensy based on the results of the test.

A vital recommendation for this project is to give telemetry data the priority in the implementation order of functions, since it would mean that all the functions can be tested swiftly with the correct graphs.

Another recommendation could be to integrate the entire system in the Paparazzi Chimera Board by implementing Dshot on Genom3 instead of implementing a serial connection using Genom with Dshot implemented on extra hardware (Teensy).

Finally, a safety feature can be added that limits the maximum current drawn by the motor. This can be done by constantly checking the current and whenever the maximum allowed current is reached, the velocity of the motor can be reduced until the current reaches back a safe level below the maximum allowed level.

# References


[1] T. E. of Encyclopaedia Britannica, "Joseph-michel and jacques-Étienne montgolfier," 2018. [Online]. Available: https://www.britannica.com/biography/Montgolfier-brothers

[2] S. E. Martínez, "Study and suppression of vibrations in rotary-wing uavs," 2015. [Online]. Available: https://www.researchgate.net/figure/Number-of-UAV-related-publications-per-year-Source-google-scholar_fig1_283703982

[3] "Dshot fpga code module," 2021. [Online]. Available: https://www.speedgoat.com/products/dshot

[4] "Kiss esc." [Online]. Available: https://www.flyduino.net/en_US/shop/product/pr1941-kiss-esc-2-5s-24a-race-edition-32bit-brushless-motor-ctrl-2715

[5] A. Mallet, C. Pasteur, M. Herrb, S. Lemaignan, and F. Ingrand, "Genom3: Building middleware-independent robotic components," *IEEE International Conference on Robotics and Automation*, pp. 1–7, 2010.

[6] "Teensy® 4.0 development board." [Online]. Available: https://www.pjrc.com/store/teensy40.html

[7] H. Lim, J. Park, D. Lee, and H. Kim, "Build your own quadrotor," *IEEE ROBOTICS AUTOMATION MAGAZINE*, pp. 1–13, 2012.

[8] "Brushless dc electric motor," 2020. [Online]. Available: https://en.wikipedia.org/wiki/Brushless_DC_electric_motor

[9] D. Lance, Gabrie, J. Meyer, and F. du Plessis, "Brushless dc motor characterisation and selection for a fixed wing uav," *IEEE Africon*, pp. 1–6, 2011.

[10] "How brushless motor and esc work," 2020. [Online]. Available: https://howtomechatronics.com/how-it-works/how-brushless-motor-and-esc-work/

[11] "What are oneshoot, multishot and dshot," 2017. [Online]. Available: https://team-blacksheep.freshdesk.com/support/solutions/articles/4000100133-what-are-oneshoot-multishot-and-dshot-

[12] D. Ribeiro, "Esc firmwares and protocols," 2017. [Online]. Available: https://fpvsampa.com/esc-firmwares-and-protocolos/

[13] "Gazebo." [Online]. Available: http://gazebosim.org/

[14] "telekyb3." [Online]. Available: https://git.openrobots.org/projects/telekyb3?jump=gollum

[15] P. Brisset, A. Drouin, M. Gorraz, P.-S. Huard, and J. Tyler, "The paparazzi solution," *2nd US-European Competition and Workshop on Micro Air Vehicles*, pp. 1–16, 2006.

[16] "Odroid xu4." [Online]. Available: https://www.hardkernel.com/

[17] "Chimera/v1.00," 2017. [Online]. Available: https://wiki.paparazziuav.org/wiki/Chimera/v1.00#On-board_Sensors

[18] A. Yiğit and J. Gangloff. [Online]. Available: https://github.com/jacqu/teensyshot